Summary

Project Summary: Customer Feedback Dashboard

This document summarizes the approach, design choices, system configuration, and execution of the Customer Feedback Dashboard project, built to streamline the process of collecting and analyzing user input.

Project Architecture and Design Approach

Component	Rationale and Design Choice
Processing Engine	We chose a powerful, third-party classification service based on its reliability and speed in handling complex text analysis tasks. This service acts as the central intelligence for extracting insights.
Interface	A multi-page web application was built (using the Streamlit framework). This decision was made to logically separate the public-facing data input from the restricted administrative processing area.
Code Structure	All business logic, including data persistence (reading/writing reviews) and calls to the external classification service, was isolated into a single `utils.py`module. This modularity makes the code easy to maintain and allows for future database integration.
Output Standardization	The analysis routine was configured to return results in a fixed JSON format. This standardization is essential for the software to automatically and reliably pull out the classification data without manual parsing errors.
Security	All access credentials for the external service were stored securely outside the source code using the deployment platform's built-in secrets manager, preventing accidental exposure on the public repository.

Task 1: User Dashboard (Data Input)

System Behavior

The User Dashboard is the project's public data collection point. It features a straightforward form that minimizes friction for the user:

- **Function:** Accepts a **Reviewer Name** and **Feedback Text**.
- **Execution:** Upon submission, the system validates the input (e.g., checks for empty fields) and then calls the data management utility to **append the new review** to the persistent data log (CSV file).
- **Data Status:** Every new review is logged with a '**Pending**' status, signaling that it is ready for administrative processing.

Task 2: Admin Dashboard (Processing and Insight Extraction)

This is the core business intelligence component, where raw feedback is transformed into actionable data.

Iteration Phase	**Objective and Strategy**	**Result**
Initial Rule Set.	Basic instruction: "Analyze this text and classify the sentiment."	Output was inconsistent and often contained extra, unstructured narrative that the software couldn't reliably parse
Standardization	**Enforced JSON Format:** Imposed a strict rule that the output must be a JSON object with only the keys: `Sentiment and Insight.` This made the output predictable.	The system could now automatically process results, but the generated insights were often too general (e.g., "Improve app performance").
Final Specification	**Actionability Constraint Added:** The rules were refined to specifically demand that the **'Insight'** be a brief, high-value, **action-oriented task** for a product manager	**Optimal Result.** Achieved reliable classification and highly useful, actionable data suitable for direct integration into a business report.

System Execution

- **Trigger:** An administrator clicks the **"Run AI ANALYSIS FOR PENDING REVIEW"** button.
- **Workflow:** The system scans the data log for all records marked **'Pending'**.
- **Processing:** For each pending record, the raw text is sent to the external classification service along with the final, rigorous input specification.
- **Update:** The system parses the returned JSON, updates the record's columns with the extracted **Sentiment** and **Actionable Insight**, and changes the **Status to 'Processed'**.
- **Resilience:** Built-in error handling manages connection failures or malformed responses, marking those records as **'Error'** for manual review rather than crashing the batch process.

This system provides a robust and efficient solution for converting unstructured customer data into structured, ready-to-use business intelligence.